计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2019,28(10):74−79 [doi: 10.15888/j.cnki.csa.007094]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Kafka 和 Storm 的车辆套牌实时分析存储系统①

任培花, 苏　铭

(山西大同大学 计算机与网络工程学院, 大同 037009)
通讯作者: 任培花, E-mail: rphren@qq.com

摘　要: 城市机动车数量、出行量的增加, 使得车辆套牌现象屡禁不止. 交管部门为了解决套牌监测的难题, 采用传统的识别方式 (如基于人工识别、基于牌照识别、基于射频识别等). 然而面对海量的日志记录, 这些方式普遍存在效率低、实时性差的问题. 为此引入大数据技术, 提出一个基于 Kafka 和 Storm 的车辆套牌实时分析存储系统. Kafka 可以作为中间件进行缓存, 提高数据采集和数据分析的同步性, 还能避免数据丢失; Storm 框架可以实现日志信息的实时计算, 然后将套牌车辆信息存入指定文档中. 整个系统具有实时、分布式存储、稳定、可扩展等特性.

关键词: Kafka; Storm; 车辆套牌; 实时分析存储

引用格式: 任培花,苏铭.基于 Kafka 和 Storm 的车辆套牌实时分析存储系统.计算机系统应用,2019,28(10):74–79. http://www.c-s-a.org.cn/1003-3254/7094.html

Real-Time Analysis Storage System for Fake Plate Vehicle Based on Kafka and Storm

REN Pei-Hua, SU Ming

(School of Computer and Network Engineering, Shanxi Datong University, Datong 037009, China)

Abstract: With the increase of the number of motor vehicles and vehicle traffic in the city, the phenomenon of fake plate vehicle appears repeatedly. In order to solve the problem of fake plate monitoring, the traffic management departments adopt traditional identification methods, such as manual identification, license plate recognition, radio frequency identification, etc. Nevertheless, facing the massive log records, these methods generally have problems of low efficiency and poor real-time performance. So big data technology was introduced, and fake plate vehicle real-time analysis storage system based on Kafka and Storm was proposed. Kafka can be used as a middleware for caching, improving the synchronization of data collection and data analysis, and avoiding data loss. The Storm framework can realize real-time calculation of log information, and then store the information of the fake plate vehicles in the specified document. The entire system has real-time, distributed storage, stability, scalability, and so on.

Key words: Kafka; Storm; fake plate vehicle; real-time analysis storage

“套牌车”是指未在交管部门办理手续, 伪造、冒用他人合法车辆和行驶驾照的车辆[1]. 近年来, 随着机动车数量的增加, 车辆套牌现象屡见不鲜, 严重损害了交通参与者的合法权益, 增加了交管部门的工作难度, 甚至增添了社会不稳定因素等. 因此, 套牌行为已成为当前交通和治安管理中的一个难点问题.

① 基金项目: 山西省高等学校教学改革创新重点项目 (J2017093/2017); 山西省高等学校教学改革项目 (2015090/2015); 大同市科技局软科学项目 (2016120/2016)
Foundation item: Major Project for Education Reform Innovation of Higher Education of Shanxi Province (J2017093/2017); Education Reform Project of Higher Education of Shanxi Province (2015090/2015); Soft Science Project of Science and Technology Bureau of Datong City (2016120/2016)
收稿时间: 2019-03-12; 修改时间: 2019-04-04, 2019-04-15; 采用时间: 2019-04-19; csa 在线出版时间: 2019-10-15

现今城市的各个重要路口, 均有车辆监控系统可以记录车牌号码、经过地点、时间、现场图片等. 传统的车辆套牌监测大多基于车辆监控系统的日志信息采用人工识别、牌照识别、或射频识别等方法识别套牌车辆. 然而伴随车辆数量和出行量地持续增加, 城市交通采集范围在逐步扩大, 按现有监控系统, 每天会产生以亿为单位的日志记录, 不管用哪种监测方法面对海量交通数据集, 必然会有成本高、效率低、实时性差等问题. 因此, 为了解决这些难题, 国内外很多人研究将大数据技术引入到车辆套牌稽查系统中, 对这些数据进行实时分析和存储.

1 相关研究

随着并行计算框架的产生, 关于车辆套牌的大数据研究正在兴起, 很多学者专家纷纷展开相关研究. 以“套牌”、“大数据”为检索词, 从 2014–2019 年期间, 对中国知网、万方数据库、维普网公开发表的有关车辆套牌大数据文章进行检索. 发现车辆套牌监测的研究主要停留在监测方法方面, 大数据技术在车辆套牌方面的应用研究不足, 采用的大数据分析框架也比较老. 另外, 研究内容主要集中在数据分析方面, 数据存储方面的研究很少.

代表性的如文献[1]提出了一种基于历史车牌识别数据 (ANPR) 集的套牌车并行检测方法 TP-Finder, 实现了基于整数划分的数据分块策略, 可准确呈现所有疑似套牌车辆的历史行车轨迹. 文献[2]提出一种基于路段阈值表和时间滑动窗口的套牌计算模型, 可以实时甄别交通数据流中的套牌嫌疑车. 文献[3]将实际车辆记录迁移到 Hadoop 集群的 HBase 中, 然后从 Hive 从 HBase 中获取同一车牌号码的时空分布情况, 通过校正因子获取最终的嫌疑套牌车. 文献[4,5]提出一种针对大规模数据集的分布式计算模型 MapReduce. 文献[6]提出一种新的基于 Hadoop 的 MapReduce 算法模型, 可以有效地解决处理海量数据时面临的性能瓶颈问题, 该算法通过引入多台硬件计算资源协同处理大规模数据下的套牌车检测. 文献[7]提出一种比 Hadoop 实时性好的分布式实时计算框架 Storm. 该框架具有健壮性、容错性、动态调整并行度等特性.

通过对文献研究可知, 目前普遍做法将实际的海量数据导入或并行连接到大数据平台上, 然后再进行数据分析, 进而监测套牌车辆. 但这些做法的实时性普遍不足, 如文献[1–3]均采用对历史记录的分析, 车辆运行每时每刻在发生, 这种做法显然实时性很差. 文献[4–6]中虽然考虑了实时性, 但 MapReduce 属于批处理算法, 等数据集到一个量的时候才启动, 势必会有一个时间延迟. 因此, 本文充分考虑车辆套牌监测的实时性要求, 借鉴文献[7], 引入 Kafka (分布式消息队列) 和 Storm (分布式实时大数据处理系统) 来解决海量车牌数据的实时分析和信息存储问题, Kafka 作为中间件可以为日志信息提供缓冲机会, 有效缓解了数据采集和数据分析的不同步问题, 提高了数据的高可用性和实时性, 避免了由于服务器故障而造成数据丢失的问题. Storm 中的运行的是拓扑 (topology) 算法相对于 MapReduce 而言, 进程是永驻的, 只要有数据就可以进行实时处理, 从而可以实现实时分析, 实现套牌监测的实时性、准确性.

2 Kafka 和 Storm 的车辆套牌实时分析存储系统分析

为了实现实时分析、存储车辆套牌信息, 本文提出一个基于 Kafka 和 Storm 的车辆套牌实时分析存储系统 (简称车辆套牌实时分析存储系统). 从系统功能模块的逻辑结构和划分两个角度分别进行描述.

2.1 逻辑层次结构

该系统包含的逻辑执行过程: 日志获取、日志缓冲、数据分析和数据存储. 图 1 是逻辑过程分层结构图.

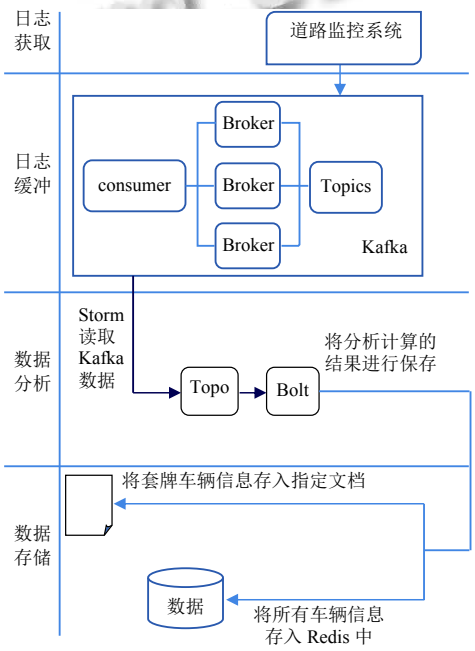

图 1 逻辑层次结构图

数据采集主要用来收集用户操作产生的车辆日志信息. 数据缓存减少了流量超峰给系统带来的压力, 该模块使用 Flume[8](分布式日志收集系统)、Kafka、Storm、Redis[9](云数据库) 等大数据框架搭建后端服务架构. 从结构图 (如图 1) 可知, 道路监控系统产生的日志信息, 先通过 Kafka 进行备份缓存, 然后将 Kafka、Storm 进行整合, 将数据导入 Storm 运行框架中. Storm 中 Spout (Storm 消息源) 会源源不断地从 Kafka 上某个主题获取数据, 并对数据进行封装发送到下游的 Bolt (Storm 消息处理者) 计算节点, Bolt 节点对数据进行实时计算, 判断是否出现套牌车辆, 算法逻辑都会在 Bolt 节点中进行运算处理. 最后系统将实时计算后的结果数据存储到服务器的文件中, 最后一次的车辆信息存入 Redis 中.

2.2 模块功能分析

车辆套牌实时分析存储系统的功能包括日志缓存与获取、日志信息切分、套牌监测和信息存储.

(1) 日志缓存与获取

通过连接道路监控系统, 在数据实时采集与数据实时处理之间搭建一个 Kafka 消息队列进行缓存, 解决数据实时采集与数据实时处理之间速度不同步和数据丢失的问题, 进一步将道路车辆监控系统产生的实时日志数据通过 Kafka 导入系统. 其次通过 Kafka 与 Storm 进行整合, 然后将数据导入 Storm 运行框架中.

(2) 日志信息切分

对采集到数据进行切分, 获取系统需要使用的有效数据.

(3) 套牌监测

通过车辆 id 从 Redis 数据库中快速读取道路车辆监控系统对应的最近监控历史记录, 计算当前车辆的区间速度, 如果车辆速度超过区间速度值就将对应车牌号码标记为套牌号牌.

(4) 信息存储

从实时的道路监控记录中提取出必要的交通信息, 并将实时交通信息存入数据库中.

2.3 类的设计

本系统涉及的主要实体类包括: KafkaTopo、SplitBolt、SpeedBolt、StorageBolt、JedisPoolUtils. 下面是整个系统的类图结构, 如图 2 所示.

实体类介绍:

(1) KafkaTopo 类主要是将 Kafka 与 Storm 进行整合, 这个类既具备缓冲特点又具备实时计算的特点.

(2) JedisPoolUtils 类主要是编写对 Redis 进行读写的 Java 客户端代码, 读取连接池配置文件, 从而提供访问 Redis 的接口.

(3) SplitBolt 类主要是对收到的日志信息进行拆分, 提取有用信息, 发送到 SpeedBolt.

(4) SpeedBolt 类主要是对 SplitBolt 发送来的信息与 Redis 中的信息进行对比, 计算出动态车速, 并且检测出套牌车辆, 将套牌车辆信息发送到 StorageBolt.

(5) StorageBolt 类主要是对 SpeedBolt 发送来的套牌车辆信息进行存储.

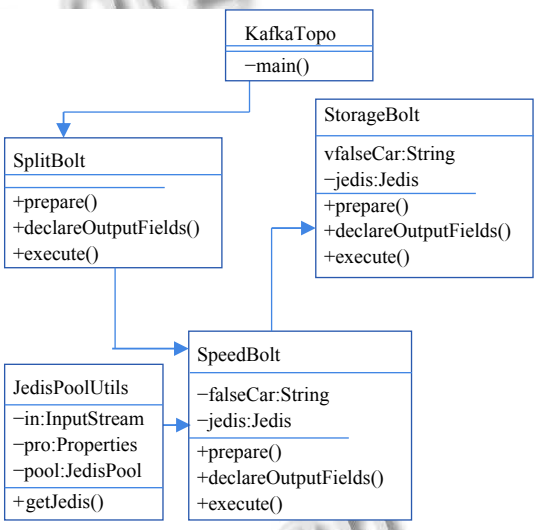

图 2 系统类图

KafkaTopo 类是整个系统的主要控制部分, 通过将数据分析与计算流程串联起来; 首先是设置 Kafka 的主题与配置其 Broker 的主机地址, 通过配置可以将日志数据获取进来; 接下来设置 3 个 Bolt 组件, 分别完成数据切分、动态车速计算与数据存储的功能; 最后将整个工程打成 jar 包提交到配置好的服务器上, 来完成实时的车辆套牌检测功能. 其具体的功能时序图如图 3 所示.

3 Kafka 和 Storm 的车辆套牌实时分析存储系统实现

本系统实现按照日志缓存与获取、日志信息切分、套牌监测和信息存储 4 个功能展开.

3.1 日志缓存与获取

日志缓存与获取是通过 Kafka 来实现, 首先将 Kafka 分别安装到 3 台 Linux 服务器上, 由于 Kafka 的高可

用是通过 Zookeeper[10]来实现的, 因此需要在 3 台服务器上安装 Zookeeper 集群, Zookeeper 集群的 IP 将作为 Kafka 的 Broker 的配置地址. 接下来通过 Storm 提供的 SpoutConfig 来将 Broker 的主机地址、Kafka 的主题、Zookeeper 集群的服务器地址、Storm 的 SpoutId 进行设置, 从而将整个数据的接收所需要依赖的环境搭建起来. 接下来对 Storm 从 kafka 获取数据的方式进行设置, 将其设置成从数据流的起始位置开始读取数据; 这些配置设置完成后, 最终实例化一个 Topology Builder 对象来将之前的配置信息进行整合, 从而使其成功获取从车辆监控系统获取的数据.

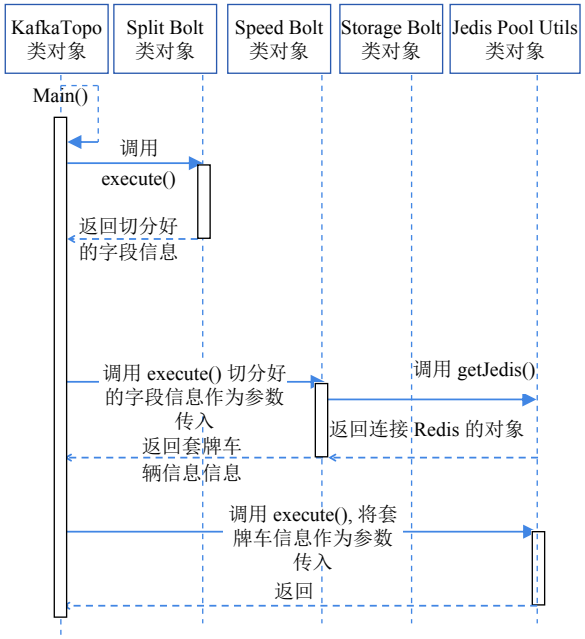

图 3 系统时序图

整个数据流的缓存是通过 Kafka 来实现的. 因为 Kafka 的 broker 结点上会有消费者机制与生产者机制, 通过设置生产者所生产消息的长度来控制消费者的消费. 使得整个数据流经过 Kafka 都会有一个缓存的过程, 如果数据出现丢失, 也可以通过设置 ACK 来实现回滚, 使得消息不会产生丢失. 整个 Kafka 的机制实现了日志信息数据流传输的高可用性.

其整个获取与缓存的流程图如图 4 所示.

3.2 日志信息的切分

通过从 Storm 的 Spout 将信息发送到指定的 SplitBolt 进行切分字段, SplitBolt 继承 BaseBasicBolt, 通过对其 execute() 方法进行重写来实现日志信息的切分. 通过其方法的 Tuple 参数接收到 Spout 发送来的日志信息, 将接收到的信息通过 toString() 方法转换成字符串类型, 从而进一步使用 split() 方法来对整个字符串信息进行切分, 获取车辆的 id、车牌、坐标、行进方向、拍摄时间等字段信息. 之后调用参数中 Collector 对象的 emit() 方法来将切分出来的字段信息进行封装, 发送到下一个 Bolt.

整个日志信息切分过程时序图如图 5 所示.

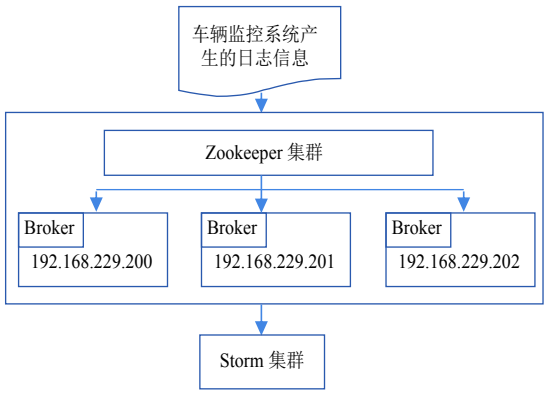

图 4 日志信息获取与缓存流程图

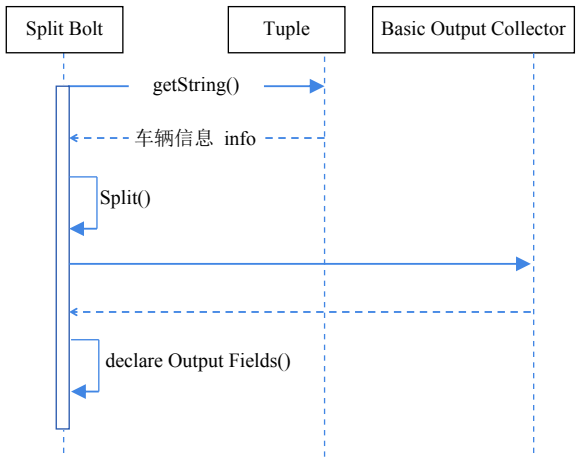

图 5 日志信息切分时序图

3.3 套牌监测

文献[1]中提出一种利用车辆时空矛盾关系判断套牌的算法, 借鉴该算法. 本文认为正常行驶的车辆在区间内的动态车速在一个限速范围内, 如果某辆车出现了套牌情况, 必然会出现在相同时间点, 坐标位置不同的情况, 而且算出的区间动态车速远高于标准车速, 因此可断定该车辆为可疑套牌车辆.

算法步骤:

Step 1. 创建 Jedis 客户端, 配置相关数据, 建立连接池, 连接 Redis 数据库接口.

Step 2. 获取当前车辆 id、车牌、坐标、运行方向、获取时间等信息

Step 3. 从 Redis 中通过 id 获取车辆信息, 若有, 拿出来通过动态车速对比, 看是否发生套牌. 若没有, 将信息存入 Redis 中

其主要业务逻辑是: 从上一 Bolt 中获取“id”, “registId”, “hangId”, “x”, “y”, “dir”, “time”和“info”信息, 通过 id 判断 Redis 中是否有该车辆信息, 若没有, 则将该车辆整条信息保存进入 Redis 中; 若有, 则将车辆信息拿出来切分, 从而获取“x”, “y”, “dir”, “time”等字段信息, 将两次的信息进行对比计算出动态车速; 若此车速远大于区间车速, 则该车为套牌车辆.

(1) 动态车速计算

从上一 Bolt 中获取车辆信息, 通过车辆 id 判断 Redis 中是否有该车辆信息, 若没有, 则将该车辆整条信息保存进入 Redis 中; 若有, 则将车辆信息拿出来切分, 从而获取坐标、时间等字段信息, 将两次的信息进行对比计算出动态车速; 若此车速远大于区间车速, 则该车为套牌车辆.

具体的业务处理是在 execute() 方法中, 通过其 Tuple 参数来接收 SplitBolt 发送过来的各个字段信息和车辆的整条 info 日志信息. 通过创建 carInfo、hangId、x、y、dir、time 等字段来获取车辆信息; 之后通过调用 Jedis 对象的 get() 方法, 将 last_hangId 传入, 看返回结果是否为空, 若可以获取到车辆信息, 则将获取到的车辆信息进行再次切分, 拿到上次记录到的字段信息, 分别设置为 last_x、last_y、last_dir、last_time; 之后调用 String 对象的 equals() 方法来判断相同车辆两次的行驶方向是否相同, 若相同并且为 x 方向, 则将 x 于 last_x 相减取绝对值, 其结果就是车辆的行驶路径; 将两次时间相减取绝对值并进行单位换算, 从而获得时间. 路程与时间相除即可得到车辆的动态车速. 若行驶方向为 y, 计算方法相同. 最后将车辆信息存入 Redis 中, 替换掉之前的车辆信息. 若从 Redis 中获取数据为空, 则说明此车辆还没有被车辆监控系统记录过, 因此直接存入 Redis 中.

(2) 套牌判定

获取动态车速后, 接下来就是与此路段的区间标准车速进行比较, 若获取的动态车速远大于区间车速, 则怀疑出现了两辆相同车牌的车辆, 将其判定为套牌车辆. 并将此车辆信息通过调用 Collector 的 emit() 方法将套牌车辆信息发送出去. 最后仍然要调用 declare OutputFields() 方法, 来指定发送字段为 info, 即套牌车辆信息.

对以上整个车速计算和套牌车的检测所做出的功能时序图如图 6 所示.

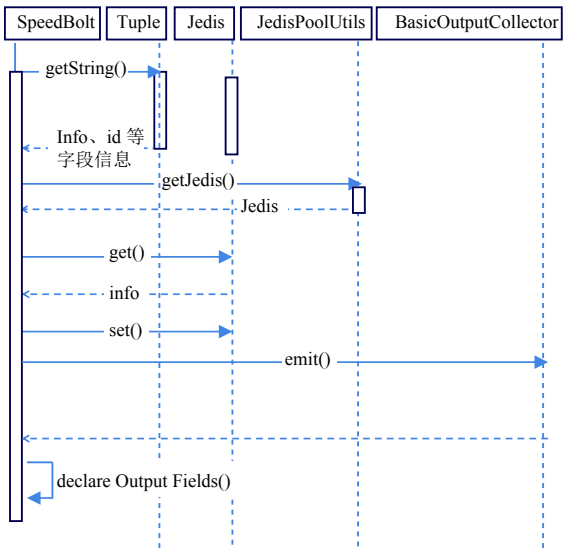

图 6 套牌监测功能时序图

3.4 信息存储

将检测出的套牌车辆信息通过编写输出代码和修改相关配置文件, 存入分布式服务器上指定的文档中, 从而可以完成信息的分布式存储.

首先, 通过 Tuple 对象获取 SpeedBolt 发送过来的套牌车辆信息, 调用 FileWriter 的 write() 方法, 来对日志信息进行写操作; 然后调用其 flush() 方法来刷新数据流, 从而使得之前写入的数据能完整输出到指定的文件中. 信息存储的功能时序图如图 7 所示.

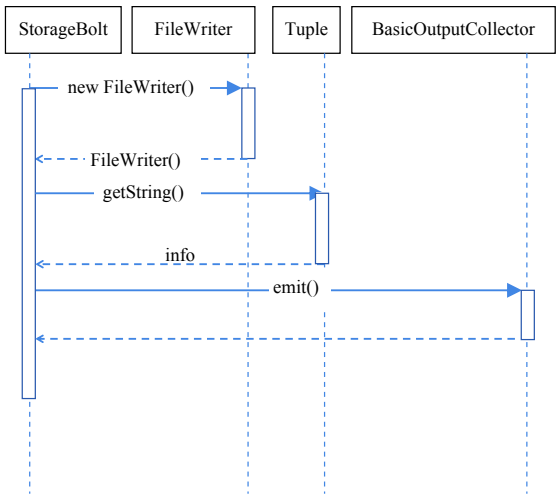

图 7 套牌车辆信息存储时序图

通过对 4 个功能模块的实现过程叙述与部分功能时序图的介绍, 对系统的整个功能进行了详细的实现. 对接收到的日志信息进行了详细的分析与计算, 整个系统的业务逻辑实现完毕, 接下来工程的部署模块可以直接将工程打成 jar 包来进行实时的运行. 至此, 系统实现已全部完成.

4 实验

本系统的实验环境包括集群搭建和数据库服务器安装, 主要包括 Zookeeper 集群、Kafka 集群、Storm 集群、Redis 安装等. 实验环境配置完毕, 将工程打成 jar 包上传到搭建好的集群运行, 通过连接车辆模拟系统产生实时数据, 可以看见 Redis 数据库上存储所有收集到的车辆信息, 如图 8 所示.

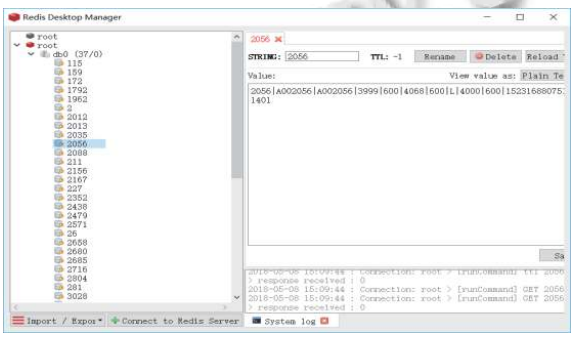

图 8 Redis 测试数据图

/home/hadoop/stormoutput 目录下存储有套牌车辆嫌疑的车辆信息, 如图 9 所示. 以这些信息为基础, 交管人员只需后期和车辆具体核实, 即可按照相关交通法规进行处理.

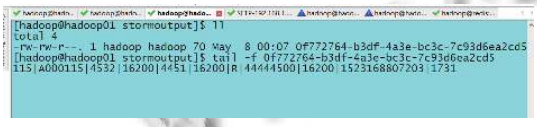

图 9 套牌车辆信息图

5 总结

针对当前海量车辆数据套牌检测实时性差的难点, 引入了大数据技术, 经过文献对比研究, 发现 Kafka 作为中间件进行缓存, 不仅保证了数据采集的效率而且还保证了数据的高可用性. 再加上, Storm 比 Hadoop 实时性处理能力强, 所以 Kafka 结合 Storm 提高了车辆套牌监测的实时性. 最后通过集群环境的搭建, 实验分析发现系统性能已达到设计目标. 在今后的研究工作中, 将探索研究套牌识别的精准度, 进而更有效地帮助交管部门完成套牌识别工作.

参考文献

1 李悦, 刘晨. 基于历史车牌识别数据的套牌车并行检测方法. 计算机应用, 2016, 36(3): 864–870. [doi: 10.11772/j.issn.1001-9081.2016.03.864]
2 乔通, 赵卓峰, 丁维龙. 面向套牌甄别的流式计算系统. 计算机应用, 2017, 37(1): 153–158. [doi: 10.11772/j.issn.1001-9081.2017.01.0153]
3 俞东进, 平利强, 李万清, 等. 一种基于 Hadoop 的套牌车识别方法: CN, 104035954A. 2014-09-10.
4 Dean J, Ghemawat S. MapReduce: Simplified data processing on large clusters. Proceedings of the Sixth Conference on Symposium on Operating Systems Design & Implementation. San Francisco, CA, USA. 2004. 10.
5 Li JJ, Wang J, Lyu B, *et al*. An improved algorithm for optimizing MapReduce based on locality and overlapping. Tsinghua Science and Technology, 2018, 23(6): 744–753. [doi: 10.26599/TST.2018.9010115]
6 王涛, 王顺, 沈益民. 交通流大数据中的套牌车并行检测算法. 湖北工程学院学报, 2014, 34(6): 29–32. [doi: 10.3969/j.issn.2095-4824.2014.06.006]
7 杨航, 朱永利. 基于 Storm 的局部放电信号集合经验模态分解. 计算机工程与应用 (网络出版). 1–8. http://kns.cnki.net/kcms/detail/11.2127.TP.20190408.1724.002.html, 2019-04-11.
8 周波. 一种基于 Flume 的海量数据分流方案. 电信科学, 2016, 32(S1): 220–225.
9 彭灿华. Redis 在高速缓存系统中的序列化算法研究. 现代电子技术, 2017, 40(22): 122–124.
10 刘芬, 王芳, 田昊. 基于 Zookeeper 的分布式锁服务及性能优化. 计算机研究与发展, 2014, 51(S1): 229–234.